

Renovating DDX

Egbert Eich

eich@suse.de
FOSDEM 2006

Why restrict us of video drivers to X?

- There may be other consumers for a video driver
 - XGI
 - Standalone DRI
 - kernel?
 - ...
- Currently driver infrastructure is married intimately with the Xserver
 - We expose X screens all the way down to the driver
 - bad idea
 - > heads cannot be migrated to different screens
 - > doing things like twin view requires ugly kludges
 - > driver cannot be used outside of an Xserver environment
 - > Testing of a driver requires to start an entire Xserver

Why restrict us of video drivers to X?

- Most data is collected at server startup time
 - modification of the data during the lifetime of a server is not modifiable
 - No graphics device hot plugging
 - No mode list changes: No display hotplugging
- All data collected during a server startup gets lost when terminating the server:
 - We need to recollect all the data!

Move driver infrastructure out of X!

- Requires a generic API between the driver and the rest of X
- Make Xserver passive to mode selection:
 - set a video mode and put X on top of it.
 - make the Xserver adapt to video mode changes
- Benefit:
 - no screen flickering when switching between console and different Xserver
 - kernel can continue to dump error to the screen even when X is running.

What do we have to look into?

- DDX: driver structure
- Common infrastructure:
 - Mode setting
 - Hardware interfacing
 - > PCI infrastructure
 - > Resource access
 - > Resource availability/sharing
 - Access to BIOS ROM
 - > Data
 - > Int10
 - > VBE

Structure

- Put different subsystems that will live in independent modules:
 - PCI subsystem
 - Resource access subsystem
 - Int10 subsystem
 - Mode selection subsystem
- Allows to test subsystems rather independently.
- Allows possible reuse of different subsystems in other software
- Forces us to design sane interfaces between different subsystems
- We can integrate support for OS specific features without affecting everybody

Fix DIX

- DIX provides infrastructure for hardware differences!
 - output device specific functions into ScreenRec structure.
 - prevents us from adding additional screen resources
 - use multiple output devices for the same screen
 - migrate between different output devices for the same screen
- Move hardware specifics completely to DDX
 - Create a DIX screen / DDX device mapping layer in DDX
 - root visual should still represent the native depth of the hardware

Configuration

- Make configuration 'on-the-fly'
 - create a configuration mechanism independent form the underlying communication interface
 - create a communication channel between config app and driver
 - > could be thru an X extension (redesigned RandR) but other mechanisms are also possible.
 - Configurable features are changing rapidly
 - > create a 'registry' for well known configuration properties
 - > provide all information to create a meaningful GUI if this information doesn't exist
 - > Handle all semantics inside the 'consumer'. GUI app should not have to have knowledge of setting interdependencies

PCI interface

- Outdated cruft: PCI Tag
- resembles data strcuture in PCI CFGMECH1 on PC hardware
- Device scanning takes ages: we check for every possible device ID on every possible bus
- Most operating systems provide all this information at almost no cost.
 - > Take advantage of this information if available
 - > Move the current device separation code to a legacy OS helper layer so that those who still need to rely on this can use it.
- Device support info stays on driver:
 - > How do we map drivers to devices?